# Dispensa di Esercizi in C++

## Fondamenti di Informatica

# Indice

## Esercizio sul Complemento a 2

Dato il seguente numero binario in complemento a 2, dire quale è il numero intero corrispondente:

11001010

Soluzione:

-128+64+8+2=-54

## Esercizio sul Complemento a 2

Convertire il seguente numero intero in binario complemento a 2 su 8 bit

-1

Soluzione:

11111111

## Esercizio sulla Codifica degli Interi

Convertire il seguente numero binario in intero, supponendo che sia espresso sia in complemento a 2 sia in modulo e segno

10110011

Soluzione:

-77 (complemento a 2)
-51 (modulo e segno)

## Esempio di Istruzioni di I/O

```
#include <iostream.h>

double x;
char c;

void main()
{
  cout << "Inserisci un Numero  " << endl;
  cin >> x;
  cout << "Il numero inserito e' " << x << endl;
  cout << "Inserisci un Carattere  " << endl;
  cin >> c;
  cout << "Il carattere inserito e' " << c << endl;
}
```

## Esempio sull'Operatore ?

```
#include<iostream.h>

int x,y;

void main()
{
  cin >> x;
  cin >> y;
  cout << ((x>y) ? x : y)<<endl;
  cout << (x>y) ? x : y;

}
```

## Esercizio sull'Operatore ?

Si consideri il seguente programma. Dire che cosa viene visualizzato e per quali valori di x e y.

```
#include<iostream.h>
#include<stdio.h>

long double x,y;
int i,n;

void main()
{
  cin >> x>>y;
  for (i=0, n=(x=y) ? x-y : y; i<n; i++)
      cout << i <<endl;
  getchar();
}
```

Soluzione:
non verra' visualizzato mai niente per qualunque valore di x e y

## Esercizio sull'Operatore &&

Dire quali valori vengono visualizzati sul video alla fine del seguente programma:

```
#include<iostream.h>
#include<stdio.h>

int a=1, x, z;

void main()
{
  cout << (z=0 && x!=4 && a++)<<endl;
  cout << (a<2 && x!=(4 && z>-1)) << endl;
  getchar();
}
```

Soluzione:

0
1

## Esempio sugli Operatori && e ||

```
#include<iostream.h>

int x,y,z,k;

void main()
{
  cin >> x;
  cin >> y;
  cin >> z;
  cin >> k;

  cout << (x>y && z<=k)<<endl;
  cout << (x>y || z<=k)<<endl;
  cout << (x<=y && z>k || x)<<endl;
  cout << (x<=y && (z>k || x))<<endl;
 }
```

## Esempio sull'Operatore ++

```
#include<iostream.h>

int x,y,z;

void main()
{
  cin >> x;
  cout << ++x<<endl;
  cout << x<<endl;
  cout << x++<<endl;
  cout << x<<endl;

  cin >> y;

  z=++y==3;
  cout << "y= "<<y << endl;
  cout << "z= " << z<<endl;

  z=y++==3;
  cout << "y= "<<y << endl;
  cout << "z= " << z<<endl;

 }
```

## Esempio sugli Operatori =, ==

```
#include<iostream.h>
#include<stdio.h>

int a,b,c;

void main()
{
  cin >> a;
  cin >> b;
  cin >> c;
  cout << "a + (b=c) " << (a+(b=c)) << endl;
  cout << "a + (b==c) " << (a+(b==c)) << endl;
  getchar();
}
```

## Esempio sull'Operatore +=

```
#include<iostream.h>
#include<stdio.h>


int x,y,x_old;

void main()
{

  cin >> x;
  cin >> y;
  x_old=x;
  x+=x+y;
  cout << x <<endl;
  x=x_old;
  x+=(x+y);
  cout << x << "   E' lo stesso !!! "<<endl;
  getchar();
}
```

## Esempio sull'Operatore /  e sul Casting

```
#include<iostream.h>
#include<stdio.h>

int x,y;

void main()
```

```
{
  cin >> x;
  cin >> y;
  cout << x/y<<endl;
  cout << (double)x/y<<endl;
  getchar();
}
```

## Esempio sul Casting e sull'Istruzione If

```
#include <iostream.h>

char c;

void main() {
 cout << "Inserisci un carattere ";
 cin >> c;

 if (c>='a' && c<='z') {
              cout << "Il carattere e' in minuscolo  " << endl;
              cout << "Il maiuscolo e' " << (char) ('A'+c-'a');
 } else if (c>='A' && c<='Z') {
              cout << "Il carattere e' in Maiuscolo  " << endl;
              cout << "Il minuscolo e' " << (char) ('a'+c-'A');
          } else cout << "Non hai inserito un carattere " << endl;
}
```

## Esempio sull'Istruzione If

```
#include<iostream.h>
#include<stdio.h>
int x;

void main()
{
  cin >>x;

  if (x<=5)
     if (x>=2) cout << "Numero compreso tra 2 e 5 "<<endl;
     else cout << "Numero minore di 2"<<endl;

  if (x<=5)
  {
     if (x>=2) cout << "Numero compreso tra 2 e 5 "<<endl;
  } else cout << "Numero maggiore di 5"<<endl;
  getchar();
}
```

## Esempio sull'Istruzione If

```
#include<iostream.h>
#include<stdio.h>
int x;

void main()
{
  cout << "Inserisci un numero diverso da 0 ";
  cin >>x;

  if (!x)
     cout << "Numero uguale a zero : Errore "<<endl;
  else
  {
     cout << "L'inverso del Numero Inserito e' "<<1/x<<endl;
     cout << "L'inverso del Numero Inserito e' "<<1.0/x<<endl;
  }
   getchar();
}
```

## Esempio sul Cattivo Uso dell'Istruzione If

```
#include<iostream.h>
#include<stdio.h>
int x;

void main()
{
   cout << "Inserisci un numero  ";
   cin >>x;

   if (x==3)
      cout << "Numero uguale a tre "<<endl;
      x++;
   cout << "Il nuovo valore di x e' "<< x<<endl;

   cout << "Inserisci un numero  ";
   cin >>x;

   if (x==3) {
      cout << "Numero uguale a tre "<<endl;
      x++;
    }
   cout << "Il nuovo valore di x e' "<< x<<endl;
   getchar();
}
```

## Esempio sul Cattivo Uso dell'Istruzione If

```
#include<iostream.h>
#include<stdio.h>
int x;

void main()
{
  cout << "Inserisci un numero  ";
  cin >>x;

  if (x==3) ;
     cout << "Numero uguale a tre "<<endl;
  getchar();
}
```

## Esempio sul Cattivo Uso dell'Istruzione If

```
#include<iostream.h>
#include<stdio.h>
int x;

void main()
{
  cout << "Inserisci un numero  ";
  cin >>x;

  if (x==3)
    cout << "Numero uguale a tre "<<endl;
    x++;
  else cout << "Numero diverso da tre "<<endl;
  getchar();
}
```

## Esempio di Ciclo While: Calcolo della potenza

```
#include<iostream.h>
#include<stdio.h>

long double base, potenza;
unsigned long int esponente;
char c;

void main()
{
```

```
  while (1) {
     cout << "Inserisci la base ";
     cin >> base;
     cout << "Inserisci l'esponente (maggiore o uguale a 0) ";
     cin >> esponente;
     potenza = 1;

     while (esponente-->0) potenza*=base;

     cout << "La potenza e' "<< potenza<<endl;

     cout << "Continui ? ";
     cin >> c;
     if (c=='n' || c=='N') break;

  }
  getchar();
}
```

## Esempio di Ciclo While: Scomposizione in numeri primi

```
#include<iostream.h>

unsigned long n,d;
int count;

void main()
{

  cout << "Inserisci un Numero intero ";
  cin >> n;

  d=2;

  while (n>1) {
    while (n % d!=0) d++;
    n/=d;
    count=1;
    while (n % d==0) {
       n/=d;
       count++;
    }
    cout << "\nUn numero primo e' "<< d << " ed ha potenza "<< count << endl;
  };
}
```

## Esempio di Ciclo Do While: Verifica validita' dati in ingresso

```
#include<iostream.h>
#include<stdio.h>

int n;

void main()
{


  do {
     cout << "Inserisci un valore intero compreso tra 4 e 8  ";
     cin >> n;
  } while (n<4 || n>8);

cout << “Il numero inserito e’ “<< n <<endl;
getchar();
}
```

## Esempio di Ciclo Do While: Calcolo della Media

```
#include<iostream.h>
#include<stdio.h>

long double media=0, n;
unsigned long int count=0;
char c;

void main()
{


  do {
     cout << "Inserisci un valore (intero o reale) ";
     cin >> n;
     media+=n;
     cout << "Numero di valori inseriti "<< ++count<<endl;

     cout << "Continui ? ";
     cin >> c;
  } while (c=='s' || c=='S');
  cout << "La media dei valori inseriti e' "<< media/count<<endl;

  getchar();
}
```

## Esercizio sul ciclo do-while

Scrivere un programma che riceve in ingresso una sequenza di caratteri numerici (inclusa la virgola) e calcola il corrispondente numero decimale (ad esempio se

```
inserisco la sequenza '2' '3' '5' ',' '2' '3', il porgramma mi restituisce il
numero 235,23).

#include<iostream.h>
#include<stdio.h>

unsigned int count;
long double v;
char c;

void main()
{
      do {
            cin >> c;
            if (c >='0' && c <='9' ) {
                  v=10*v + c -'0';
                  if (count) count*=10;
            } else if (c==',') count=1;

            cout << "Continui ? ";
            cin >>c;
      } while (c=='s' || c=='S');
      cout<< (float)(v/count);
      getchar();
}
```

## Esempio di Ciclo Do While: Calcolo del Fattoriale

```
#include<iostream.h>
#include<stdio.h>

long double fattoriale;
unsigned long int n;
char risposta;

void main()
{

  do {
     fattoriale=1.0;

     do {
        cout << "Inserisci un numero intero positivo (>1) di cui calcolare il fattoriale  ";
        cin >> n;
     } while (n<=1);

    do
       fattoriale*=n--;
    while (n>1);

    cout << "Il fattoriale e' "<< fattoriale <<endl;

    cout << "Continui ? ";
```

```
    cin >> risposta;
  } while (risposta!='n' && risposta!='N');
  getchar();
}
```

Valori Massimi:

unsigned long int 4.294.967.295
unsigned double ≈ E+308
unsigned long double ≈ E+4932

| n | fattoriale | Tipo di dato necessario alla codifica |
|---|---|---|
| 2 | 2.00 | (unsigned long int) |
| 11 | 39916800.00 | (unsigned long int) |
| 12 | 479001600.00 | (unsigned long int) |
| 13 | 6227020800.00 | (unsigned double) |
| 25 | 1.6E+25 | (unsigned double) |
| 160 | 4.7E+284 | (unsigned double) |
| 170 | 7.3E+306 | (unsigned double) |
| 1000 | 4.02387E+2567 | (unsigned long double) |
| 1751 | 3.67E+4920 | (unsigned long double) |
| 1754 | 6.42E+4930 | (unsigned long double) |

## Esempio di Ciclo For: Massimo tra n numeri

```
#include<iostream.h>

long double max,x;
double y;
int i,n;

void main()
{

  cout << "Inserisci il numero di numeri da leggere ";
  cin >> n;

  cout << "Inserisci un numero ";
  cin >> max;

  for (i=0;i<n-1;i++) {
          cout << "Inserisci un numero ";
          cin >> x;
          if (x>max) max=x;
  }
```

```
  cout << "Il massimo numero e' " << max << endl;
}
```

## Esercizio sul Ciclo for

```
Dire quale valore assume la variabile y alla fine del programma, ossia cosa
viene visualizzato sul video:

#include<iostream.h>
#include<stdio.h>

char c;
int y;

void main()
{
  for (c='z'; c<'a';c++)
                y++;
  cout << y<<endl;

  getchar();

}
```

## *Soluzione:*

0

## Esempio di Ciclo For: Numeri di Fibonacci

```
#include<iostream.h>
int n, fa, fb;
char c;


void main()
{
   do{
      do { cout<<"inserisci un numero >=2   ";
           cin>>n;
   }while (n<2);

  fa=0, fb=1;

   for (int i=1;i<n;i++) {
         fb+=fa;
         fa=fb-fa;
   }
   cout<<"il Numero di Fibonacci di "<<n<<"  e'     "<<fb<<endl;
   cout<<"continui [s/n]?";
    cin>>c;
```

```
    }while(c!='n'&&c!='N');
}
```

## Esempio sulle Variabili Puntatore

```
#include<iostream.h>
#include<stdio.h>

int x=14;
int *p;

void main()
{

  cout << "Indirizzo contenuto nella variabile p " << p << endl;

  p=&x;

  cout << "Indirizzo contenuto nella variabile p " << p << endl;
  cout << "Contenuto della locazione di indirizzo p e'  " << *p << endl;


  p=new int;
  *p=30;

  cout << "Indirizzo contenuto nella variabile p " << p << endl;
  cout << "Contenuto della locazione di indirizzo p e'  " << *p << endl;

  getchar();
}
```

## Esempio di Uso di Vettori Statici: Fusione di Due Vettori Ordinati

```
#include<iostream.h>
#include<stdio.h>

const int dim1=10, dim2=20, dim3=30;


double vet1[dim1], vet2[dim2], vet3[dim3];

int i,j,k;

void main(){

   cout << "Inserisci Elemento di indice " << i << " del primo vettore ";
   cin >> vet1[i];
   for (i=1;i<dim1; i++)
```

```
      do {
            cout << "Inserisci Elemento di indice " << i << " del primo vettore ";
            cin >> vet1[i];
      } while (vet1[i]< vet1[i-1]);

  i=0;
  cout << "Inserisci Elemento di indice " << i << " del secondo vettore ";
  cin >> vet2[i];
  for (i=1;i<dim2; i++)
      do {
            cout<<"Inserisci Elemento di indice " << i << " del secondo vettore ";
            cin >> vet2[i];
      } while (vet2[i]<vet2[i-1]);


   i=0;
   j=0;
   k=0;
   while (i<dim1 && j<dim2) {
            if (vet1[i]<vet2[j])
                              vet3[k++]=vet1[i++];
            else if (vet1[i]>vet2[j]) vet3[k++]=vet2[j++];
                  else {
                      vet3[k++]=vet1[i++];
                      vet3[k++]=vet2[j++];
                  }
   }
   if (i<dim1)
         while (i<dim1) vet3[k++]=vet1[i++];
   else if (j<dim2) while (j<dim2) vet3[k++]=vet2[j++];

   for (k=0; k<dim3; k++) {
      cout << "Elemento di indice " << k;
      cout  << " del terzo vettore  e'  " << vet3[k]<<endl;
   }
   getchar();
}
```

## Esempio di Allocazione Dinamica di Vettori: Fusione di Due Vettori Ordinati

```
#include<iostream.h>
double *vet1, *vet2, *vet3;
int i,j,k,dim1, dim2, dim3;

void main(){
  do {
    cout << "Inserisci la dimensione del primo vettore ";
    cin >> dim1;
  } while (dim1<=0);
```

```
  vet1=new double[dim1];

  do {
    cout << "Inserisci la dimensione del secondo vettore ";
    cin >> dim2;
  } while (dim2<=0);

  vet2=new double[dim2];

  vet3=new double[dim3=dim1+dim2];

  i=0;
  cout << "Inserisci Elemento di indice " << i << " del primo vettore ";
  cin >> vet1[i];
  for (i=1;i<dim1; i++)
     do {
           cout << "Inserisci Elemento di indice " << i << " del primo vettore ";
           cin >> vet1[i];
     } while (vet1[i]< vet1[i-1]);

  i=0;
  cout << "Inserisci Elemento di indice " << i << " del secondo vettore ";
  cin >> vet2[i];
  for (i=1;i<dim2; i++)
     do {
           cout<<"Inserisci Elemento di indice " << i << " del secondo vettore ";
           cin >> vet2[i];
     } while (vet2[i]<vet2[i-1]);


  i=0;
  j=0;
  k=0;
  while (i<dim1 && j<dim2) {
            if (vet1[i]<vet2[j])
                              vet3[k++]=vet1[i++];
            else if (vet1[i]>vet2[j]) vet3[k++]=vet2[j++];
                    else {
                        vet3[k++]=vet1[i++];
                        vet3[k++]=vet2[j++];
                    }
  }
  if (i<dim1)
         while (i<dim1) vet3[k++]=vet1[i++];
  else if (j<dim2) while (j<dim2) vet3[k++]=vet2[j++];

  for (k=0; k<dim3; k++)
     cout<<"Elemento di indice "<<k<<" del terzo vettore  "<<vet3[k]<<endl;

  delete vet1;
```

```
   delete vet2;
   delete vet3;
}
```

## Esempio di Allocazione Dinamica di Vettori: Prodotto di Due Vettori

```
#include<iostream.h>
#include<stdio.h>

double *vet1, *vet2, prodotto;
int i,dim;

void main(){

  do {
    cout << "Inserisci la dimensione comune ai due vettori ";
    cin >> dim;
  } while (dim<=0);

  vet1=new double[dim];
  vet2=new double[dim];

  for (i=0;i<dim; i++) {
            cout << "Inserisci Elemento di indice " << i << " del primo vettore ";
            cin >> vet1[i];
            cout<<"Inserisci Elemento di indice " << i << " del secondo vettore ";
            cin >> vet2[i];
   }

  prodotto=0;

  for (i=0;i<dim; i++)
            prodotto+=vet1[i]*vet2[i];

   cout << "Il prodotto dei due vettori e' "<< prodotto<<endl;

   getchar();

   delete vet1;
   delete vet2;

}
```

## Esempio di Gestione delle Matrici Statiche: Riempimento e Visualizzazione

```
#include <iostream.h>
#include <stdio.h>
```

```
const int righe=2, colonne=2;
int i,j;
int matrice[righe][colonne];
int row,column;



void main()
{

  cout << "Inserimento Matrice Statica "<<endl;

  for (i=0; i< righe; i++)
     for (j=0; j< colonne; j++){
           cout << "Inserisci Elemento " << i << j << "  ";
           cin >> matrice[i][j];
     }
  }



  cout << "Visualizzazione Matrice Statica "<<endl;

   for (i=0; i<righe; i++)
    for (j=0; j<colonne; j++)
      cout<<"Elemento di Indice " <<i<< " , " <<j<< " = " << matrice[i][j] <<endl;

  getchar();
}
```

## Esempio di Gestione Matrici Dinamiche: Inserimento e Visualizzazione

```
#include <iostream.h>
#include <stdio.h>

int i,j;
double **matrix;
int row,column;


void main()
{
  cout << "Inserisci numero di righe ";
  cin >> row;
  cout << "Inserisci il numero di colonne ";
  cin >> column;

  matrix=new double * [row];
  for (i=0; i< row; i++)
```

```
                matrix[i]=new double[column];

  cout << "Inserimento Matrice Dinamica "<<endl;

  for (i=0; i<row; i++)
     for (j=0; j<column; j++){
            cout << "Inserisci Elemento " << i << j << "   ";
            cin >> matrix[i][j];
     }


  cout << "Visualizzazione Matrice Dinamica "<<endl;

  for (i=0; i<row; i++)
    for (j=0; j<column; j++)
      cout<<"Elemento di Indice "<<i<<","<<j<<" = "<< matrix[i][j] << endl;

   getchar();

   for (i=0; i< row; i++)
              delete matrix[i];

   delete matrix;

}
```

## Esempio di Gestione Matrici Dinamiche: Prodotto di Matrici

```
#include <iostream.h>
#include <stdio.h>

int i,j,k;
double **m1, **m2, **m3;
int r1,c1,r2,c2,r3,c3;

void main()
{

  do {
   cout << "Inserisci numero di righe della prima matrice";
   cin >> r1;
  } while (r1<1);
  do {
    cout << "Inserisci il numero di colonne della prima matrice";
    cin >> c1;
  } while (c1<1);
  r2=c1;
  do {
    cout << "Inserisci il numero di colonne della seconda matrice";
    cin >> c2;
```

```
  } while (c2<1);

  m1=new double * [r1];
  for (i=0; i< r1; i++)
              m1[i]=new double[c1];
  m2=new double * [r2];
  for (i=0; i< r2; i++)
              m2[i]=new double[c2];
  m3=new double * [r1];
  for (i=0; i< r1; i++)
              m3[i]=new double[c2];



  cout << "Inserimento Prima Matrice "<<endl;

  for (i=0; i<r1; i++)
     for (j=0; j<c1; j++){
              cout << "Inserisci Elemento " << i << j << "   ";
              cin >> m1[i][j];
     }

  cout << "Inserimento Seconda Matrice "<<endl;

  for (i=0; i<r2; i++)
     for (j=0; j<c2; j++){
              cout << "Inserisci Elemento " << i << j << "   ";
              cin >> m2[i][j];
     }


  for (i=0; i<r1; i++)
    for (j=0;j<c2;j++) {
          m3[i][j]=0;
          for (k=0; k<c1; k++)
                    m3[i][j]+=m1[i][k]*m2[k][j];

    }

  cout << "Visualizzazione Matrice Prodotto "<<endl;

  for (i=0; i<r1; i++)
    for (j=0; j<c2; j++)
      cout<<"Elemento di Indice "<<i<<","<<j<<" = "<< m3[i][j] << endl;

   getchar();


   for (i=0; i< r1; i++)
              delete m1[i];
```

```
    delete m1;

   for (i=0; i< r2; i++)
                delete m2[i];

    delete m2;

   for (i=0; i< r1; i++)
                delete m3[i];

    delete m3;
}
```

## Esempio di Istruzioni di I/O per le Stringhe

```
#include <iostream.h>

char parola[30];
double x;
char c;

void main()
{
   cout << "Inserisci una stringa  " << endl;
   cin.get(parola,31,'\n');
   cout << "La parola inserita e' " << parola << endl;
   cout << "Inserisci un Numero  " << endl;
   cin >> x;
   cout << "Il numero inserito e' " << x << endl;
   cout << "Inserisci un Carattere  " << endl;
   cin >> c;
   cout << "Il carattere inserito e' " << c << endl;
   cin.ignore(80,'\n');
   cout << "Inserisci una stringa  " << endl;
   cin.get(parola,31,'\n');
   cout << "La parola inserita e' " << parola << endl;
}
```

## Esempio sulla Funzione STRCPY

```
#include <iostream.h>
#include <stdio.h>


const int dim=10;
char *p1, *p2;

void main()
```

```
{

  p1=new char[dim];
  p2=new char[dim];

  cout << "Inserisci la prima stringa ";
  cin.get(p1, dim, '\n');

  strcpy(p2,p1);
  cout << "La seconda stringa e' "<<p2<<endl;
 getchar();

}
```

## Esempio sulla Funzione STRCMP

```
#include <iostream.h>
#include <stdio.h>


const int dim=10;
char *p1, *p2;

void main()
{

  p1=new char[dim];
  p2=new char[dim];

  cout << "Inserisci la prima stringa ";
  cin.get(p1, dim, '\n');

  cin.ignore(80, '\n');
  cout << "Inserisci la seconda stringa ";
  cin.get(p2, dim, '\n');
  if (!strcmp(p2,p1)) cout <<"Le stringhe sono uguali "<<endl;
  else if (strcmp(p2,p1)<0) cout <<"La prima stringa e' piu' grande "<<endl;
       else cout << "La seconda stringa e' piu' grande"<<endl;
   getchar();

}
```

## Esempio sulla Funzione STRLEN

```
#include <iostream.h>
#include <stdio.h>


const int dim=10;
```

```
char *p1, *p2;

void main()
{

  p1=new char[dim];
  p2=new char[dim];

  cout << "Inserisci la prima stringa ";
  cin.get(p1, dim, '\n');


  cout <<"La lunghezza della prima stringa e' "<<strlen(p1)<<endl;

  cin.ignore(80, '\n');
  cout << "Inserisci la seconda stringa ";
  cin.get(p2, dim, '\n');

  cout <<"La lunghezza della seconda stringa e' "<<strlen(p2)<<endl;

 getchar();

}
```

## Esempio sulle Variabili Globali e Locali

```
#include<iostream.h>
#include<stdio.h>

int x;

void main()
{
  int y;
  cout << "Il valore iniziale di x (variabile globale) e' " << x<< endl;
  cout << "Il valore iniziale di y (variabile locale) e' " << y<< endl;
  getchar();
}
```

## Esempio sulle Variabili Statiche

```
#include<iostream.h>
#include<stdio.h>

void prova()
{
 static int x=40;
```

```
 cout << ++x << endl;
}

void main()
{

  for (int i=0; i< 4;i++)
                    prova();

  getchar();
}
```

## Esempio sulle Procedure senza parametri formali

```
#include <iostream.h>
#include <stdio.h>
long double base;
unsigned long int esponente;

void potenza();

void main()
{
     cout << "Inserisci la base ",
     cin >> base;
     cout << "Inserisci l'esponente ",
     cin>> esponente;
     potenza();
     getchar();

}

void potenza()
{
   long double p=1;

   while (esponente-->=1) p*=base;
   cout << "La potenza e' "<<p<<endl;
}
```

## Esempio sulle Procedure con parametri formali

```
#include <iostream.h>
#include <stdio.h>
long double b1,b2,b3;
unsigned long int e1,e2,e3;

void potenza(long double b, unsigned int e);
```

```
void main()
{
     cout << "Inserisci la base ";
     cin >> b1;
     cout << "Inserisci l'esponente ";
     cin>> e1;
     potenza(b1,e1);

     cout << "Inserisci la base ";
     cin >> b2;
     cout << "Inserisci l'esponente ";
     cin>> e2;
     potenza(b2,e2);

     cout << "Inserisci la base ";
     cin >> b3;
     cout << "Inserisci l'esponente ";
     cin>> e3;
     potenza(b3,e3);

     getchar();

}

void potenza(long double b, unsigned int e)
{
  long double p=1;

  while (e-->=1) p*=b;
  cout << "La potenza e' "<<p<<endl;
}
```

## Esempio sulle Funzioni con parametri formali

```
#include <iostream.h>
#include <stdio.h>
long double b1,b2,b3;
unsigned long int e1,e2,e3;

long double potenza(long double b, unsigned int e);

void main()
{
     cout << "Inserisci la base ";
     cin >> b1;
     cout << "Inserisci l'esponente ";
     cin>> e1;
     cout << "La potenza e' "<< potenza(b1,e1);
```

```
    cout << "Inserisci la base ";
    cin >> b2;
    cout << "Inserisci l'esponente ";
    cin>> e2;
    cout << "La potenza e' "<< potenza(b2,e2);

    cout << "Inserisci la base ";
    cin >> b3;
    cout << "Inserisci l'esponente ";
    cin>> e3;
    cout << "La potenza e' "<< potenza(b3,e3);

    getchar();

}

long double potenza(long double b, unsigned int e)
{
  long double p=1;

  while (e-->=1) p*=b;
  return p;
}
```

## Esempio di Procedure con Parametri Formali: Inserimento e Visualizzazione delle Matrici Statiche

```
#include <iostream.h>
#include <stdio.h>

const int righe=2, colonne=2;
int i,j;
int matrice[righe][colonne];
int row,column;


void inserimento_matrice_statica(int m[][colonne], int r, int c)
{
 int i,j;

for (i=0; i< r; i++)
      for (j=0; j< c; j++){
            cout << "Inserisci Elemento " << i << j << "  ";
            cin >> m[i][j];
      }
}

void visualizza_matrice_statica(int m[][colonne], int r, int c)
{
 int i,j;
```

```
  for (i=0; i<r; i++)
     for (j=0; j<c; j++)
       cout<<"Elemento di Indice " <<i<< " , " <<j<< " = " << m[i][j] <<endl;
}


void main()
{

   cout << "Inserimento Matrice Statica "<<endl;
   inserimento_matrice_statica(matrice, righe, colonne);

   cout << "Visualizzazione Matrice Statica "<<endl;
   visualizza_matrice_statica(matrice, righe, colonne);

   getchar();
}
```

## Esempio di Funzione Ricorsiva: Fattoriale

```
#include<iostream.h>

char risposta;
int numero;

long double fattoriale(int n)
{
   if (n>1) return(n*fattoriale(n-1));
   else return(1);
}

void main()
{
   do {
      cout << "Inserisci n ",
      cin >> numero;
      cout << "il fattoriale e' " << fattoriale(numero) << endl;

      cout << "Continui ? ";
      cin >> risposta;
   } while (risposta!='n' && risposta!='N');
}
```

Il Numero piu' grande che puo' essere immesso e' n=1754, che fornisce il valore di fattoriale(1754)=1.97926*104930

## Esempio di Funzione Ricorsiva: Ricerca Binaria

```
#include<iostream.h>
#include<stdio.h>

int i;
const int dim=10;

double vet[dim], elemento;

void inserisci(double v[], int d)
{
  int i=0;

  cout << "Inserisci Elemento di indice " << i << " del vettore ";
  cin >> v[i];
  for (i=1;i<d; i++)
  do {
        cout << "Inserisci Elemento di indice " << i << " del vettore ";
        cin >> v[i];
  } while (v[i]< v[i-1]);
}


int ricerca_binaria(double v[], double elem, int inf, int sup)
{
 int medio;

 if (inf < sup) {
     medio=(inf+sup)/2;
     if (v[medio]==elem) return(1);
     else if (v[medio]<elem) return(ricerca_binaria(v,elem,medio+1,sup));
           else return(ricerca_binaria(v,elem,inf, medio-1));
 } else return(0);
}


void main()
{
  inserisci(vet,dim);
  cout << "Inserisci elemento da cercare ";
  cin >> elemento;
  if (ricerca_binaria(vet,elemento,0,dim-1))
           cout << "Elemento trovato " << endl;
  else cout << "Elemento non esistente " << endl;

  getchar();
}
```

## Esempio di Cattivo Uso della Ricorsione: Numeri di Fibonacci

```
#include<iostream.h>
int n;
char c;

int fibonacci (int v)
{
  if (!v) return(0);
  if (v==1) return(1);
  return(fibonacci(v-1)+fibonacci(v-2));
   }

void main()
{
    do{
        do { cout<<"inserisci un numero >=2   ";
                cin>>n;
        }while (n<2);
        cout<<"il Numero di Fibonacci di "<<n<<"  e'     "<<fibonacci(n)<<endl;
        cout<<"continui [s/n]?";
        cin>>c;
    }while(c!='n'&&c!='N');
}
```

## Esempio di Allocazione Dinamica di Vettore: Ricerca Binaria

```
#include<iostream.h>
#include<stdio.h>

double *vet, elemento;
int i,dim;

void inserisci(double v[], int d)
{
   int i=0;

   cout << "Inserisci Elemento di indice " << i << " del vettore ";
   cin >> v[i];
   for (i=1;i<d; i++)
   do {
          cout << "Inserisci Elemento di indice " << i << " del vettore ";
          cin >> v[i];
  } while (v[i]< v[i-1]);
}


int ricerca_binaria(double v[], double elem, int inf, int sup)
{
```

```
  int medio;

  if (inf < sup) {
      medio=(inf+sup)/2;
      if (v[medio]==elem) return(1);
      else if (v[medio]<elem) return(ricerca_binaria(v,elem,medio+1,sup));
          else return(ricerca_binaria(v,elem,inf, medio-1));
   } else return(0);
}


void main()
{
   do {
     cout << "Inserisci la dimensione del vettore ";
     cin >> dim;
    } while (dim<=0);

   vet=new double[dim];

   inserisci(vet,dim);
   cout << "Inserisci elemento da cercare ";
   cin >> elemento;
   if (ricerca_binaria(vet,elemento,0,dim-1))
            cout << "Elemento trovato " << endl;
   else cout << "Elemento non esistente " << endl;
   getchar();

}
```

## Esercizio su Procedure, Parametri Formali e Stringhe

Scrivere una funzione o procedura che riceve come parametro formale un vettore di caratteri (STRINGA). La funzione o procedura NON RICEVE LA DIMENSIONE DEL VETTORE DI CARATTERI. La funzione o procedura calcola la lunghezza effettiva della stringa (ossia il numero di caratteri escluso il carattere '\0') e alloca un vettore di interi di dimensione pari a tale lunghezza. Infine riempie tale vettore con i codici ASCII dei caratteri della stringa ricevuta in ingresso. La funzione o procedura deve fornire in uscita il vettore così ottenuto e la sua dimensione. CIO' NON DEVE ESSERE OTTENUTO TRAMITE L'USO DI COUT, MA DEVE ESSERE UTILIZZATO IL MECCANISMO DEI PARAMETRI FORMALI VARIABILE (REFERENCE).

**Soluzione:**

```
void compito (char * v, int * & c, int & dim_c)
{
        int i;

        dim_c=0;

        for ( ; *v!='\0'; v++)
                                dim_c++;
```

```
        c=new int[dim_c];

        for (i=0; i<dim_c; i++)
                c[i]=v[i];
}
```

## Esercizio su Procedure, Parametri Formali e Stringhe

Scrivere una funzione che riceve come parametro formale una stringa di caratteri tutti minuscoli e calcola il carattere più piccolo (in ordine alfabetico). LA FUNZIONE NON RICEVE LA DIMENSIONE DELLA STRINGA COME PARAMETRO FORMALE.

Soluzione:

```
char minimo (char *s)
{
      char min='z';

      for ( ; *s!='\0'; s++)
            if (*s<min)
                        min=*s;
      return min;
}
```

## Esempio di Gestione delle Matrici Dinamiche

```
#include <iostream.h>
#include <stdio.h>

int i,j;
int **matrix;
int row,column;

void inserimento_matrice_dinamica(int **m, int r, int c)
{
  int i,j;

  for (i=0; i<r; i++)
     for (j=0; j<c; j++){
           cout << "Inserisci Elemento " << i << j << "  ";
           cin >> m[i][j];
     }
}

void visualizza_matrice_dinamica(int **m, int r, int c)
{
  int i,j;

  for (i=0; i<r; i++)
```

```
    for (j=0; j<c; j++)
      cout<<"Elemento di Indice "<<i<<","<<j<<" = "<< m[i][j] << endl;
}


void main()
{
  cout << "Inserisci numero di righe ";
  cin >> row;
  cout << "Inserisci il numero di colonne ";
  cin >> column;

  matrix=new int * [row];
  for (i=0; i< row; i++)
            matrix[i]=new int[column];

  cout << "Inserimento Matrice Dinamica "<<endl;
  inserimento_matrice_dinamica(matrix,row,column);


  cout << "Visualizzazione Matrice Dinamica "<<endl;
  visualizza_matrice_dinamica(matrix,row,column);


  getchar();

  for (i=0; i< row; i++)
            delete matrix[i];

  delete matrix;

}
```

## Esempio di Gestione delle Matrici Dinamiche

```
#include <iostream.h>
#include <stdio.h>

int i,j;
int *vettore;
int row,column;


void inserimento_matrice_dinamica(int v[], int r, int c)
{
 int i,j;

  for (i=0; i< r; i++)
     for (j=0; j<c; j++){
```

```
            cout << "Inserisci Elemento " << i << j << "  ";
            cin >> v[i*c+j];
      }
}

void visualizza_matrice_dinamica(int v[], int r, int c)
{
 int i,j;

 for (i=0; i<r; i++)
    for (j=0; j<c; j++)
      cout<<"Elemento di Indice "<<i<<","<<j<<" = "<< v[i*c+j] << endl;
}


void main()
{
  cout << "Inserisci numero di righe ";
  cin >> row;
  cout << "Inserisci il numero di colonne ";
  cin >> column;

  vettore=new int[row*column];

  cout << "Inserimento Matrice Dinamica "<<endl;
  inserimento_matrice_dinamica(vettore,row,column);

  cout << "Visualizzazione Matrice Dinamica "<<endl;
  visualizza_matrice_dinamica(vettore,row,column);


  getchar();

  delete vettore;
}
```

## Esempio di Gestione delle Matrici Dinamiche

```
#include <iostream.h>
#include <stdio.h>

int i,j;
int **matrix;
int *vettore;
int row,column;


void inserimento_matrice_dinamica(int **m, int r, int c)
{
for (i=0; i< r; i++)
```

```
      for (j=0; j< c; j++){
            cout << "Inserisci Elemento " << i << j << "   ";
            cin >> m[i][j];
      }
}

void visualizza_matrice_dinamica(int **m, int r, int c)
{
  int i,j;

  for (i=0; i<r; i++)
     for (j=0; j<c; j++)
       cout<<"Elemento di Indice "<<i<<","<<j<<" = "<< m[i][j] << endl;
}

void main()
{
   cout << "Inserisci numero di righe ";
   cin >> row;
   cout << "Inserisci il numero di colonne ";
   cin >> column;

   vettore=new int[row*column];
   matrix=new int * [row];
   for (i=0; i< row; i++)
              matrix[i]=vettore+i*column;

   cout << "Inserimento Matrice Dinamica "<<endl;
   inserimento_matrice_dinamica(matrix,row,column);

   cout << "Visualizzazione Matrice Dinamica "<<endl;
   visualizza_matrice_dinamica(matrix,row,column);


   getchar();

   delete vettore;
   delete matrix;

}
```

## Esempio di Uso della Struttura

```
#include<iostream.h>

struct persona {
          char cognome[20], nome[30];
          long int telefono;
};
```

```
void stampa(struct persona p) //si può anche non mettere struct
{
   cout << p.cognome << p.nome << p.telefono;
}

void inserisci(struct persona & p)   //si può anche non mettere struct
{
   cout << “Inserisci il Cognome “;
   cin >> p.cognome;
   cout << “Inserisci il Nome “;
   cin >> p.nome;
   cout << “Inserisci il Telefono “;
   cin >> p.telefono;

}

void main()
{
 struct persona elem;
 struct persona * pelem;

 inserisci(elem);
 stampa(elem);

pelem = new struct persona;
 cout << "\n\n";

 cout << “Inserisci il Cognome “;
 cin >> pelem->cognome;
 cout << “Inserisci il Nome “;
 cin >> pelem->nome;
 cout << “Inserisci il Telefono “;
 cin >> pelem->telefono;

 cout << "\nCognome "<<pelem->cognome;
 cout << "\nNome "<<pelem->nome;
 cout << "\nTelefono "<<pelem->telefono;

}
```

## Esempio di Uso della Struttura contenente Vettori

```
#include<iostream.h>

const int d=5000;

struct media {
        long double vettore[d];    //dimensione del vettore molto grande !!!!!
```

```
        long double average;
};

void inserisci(media &m, int dim)
{

   m.average=0;

   for (int i=0; i<dim; i++) {
        cout << "Inserisci elemento di indice " << i << "  ";
        cin >> m.vettore[i];
        m.average+=m.vettore[i];
  }
 }

long double calcola(const media &m)   //si passa per puntatore a causa delle grosse dimensioni
{
   return(m.average);

}

void main()
{

  media elem;

   inserisci(elem,d);
  cout <<" La media e' " <<calcola(elem)<<endl;
}
```

## Esercizio su: Vettore Dinamico e tipo Struct

Scrivere un programma composto da:

- una definizione di un tipo struct con campi: cognome, nome, età, città. Nel seguito tale tipo verrà chiamato **persona**.
- una funzione o procedura che alloca un vettore di puntatori al tipo persona. La funzione o procedura riceve in ingresso il vettore e la sua dimensione. La funzione o procedura, oltre ad allocare il vettore di puntatori, deve porre tutti gli elementi del vettore (ossia i puntatori al tipo persona) al valore NULL.
- una funzione o procedura che riceve come parametro formale un vettore di puntatori al tipo persona e la sua dimensione. La funzione o procedura deve allocare una variabile di tipo persona per ciascun elemento del vettore e porre in tale elemento il puntatore alla variabile struct persona appena allocata. Infine la funzione o procedura deve riempire i campi cognome, nome, età e città di ogni variabile di tipo struct persona precedentemente allocata.
- una funzione che riceve come parametro formale un vettore di puntatori al tipo persona, la sua dimensione e calcola l'età più alta tra tutte le variabili i cui puntatori sono contenuti nel vettore. La funzione deve restituire tale valore.
- una funzione o procedura che disalloca un vettore di puntatori al tipo persona.
- un main che richiama opportunamente tutte le funzioni/procedure precedenti

## *Soluzione:*

```
#include<iostream.h>
#include<stdio.h>

/*una definizione di un tipo struct con campi: cognome, nome, età, città. Nel
seguito tale tipo verrà chiamato persona.*/

struct persona {
      char cognome[20], nome[20];
      unsigned int eta;
      char citta[20];
} ** vettore;

int dimensione;

/* una funzione o procedura che alloca un vettore di puntatori al tipo persona.
La funzione o procedura riceve in ingresso il vettore e la sua dimensione. La
funzione o procedura, oltre ad allocare il vettore di puntatori, deve porre
tutti gli elementi del vettore (ossia i puntatori al tipo persona) al valore
NULL.*/

void alloca (persona ** & v, int dim) {
      v=new persona * [dim];
      for (int i=0; i<dim; i++)
            v[i]=NULL;
}

/*una funzione o procedura che riceve come parametro formale un vettore di
puntatori al tipo persona e la sua dimensione. La funzione o procedura deve
allocare una variabile di tipo persona per ciascun elemento del vettore e porre
in tale elemento il puntatore alla variabile struct persona appena allocata.
Infine la funzione o procedura deve riempire i campi cognome, nome, età e città
di ogni variabile di tipo struct persona precedentemente allocata.*/

void riempi (persona ** v, int dim) {
   for (int i=0; i<dim;i++) {
      v[i]=new persona;
      cout << "Elemento di indice "<<i<<endl;
      cout <<"Cognome = ";
      cin >>v[i]->cognome;
      cout <<"Nome = ";
      cin >>v[i]->nome;
      cout <<"Eta' = ";
      cin >>v[i]->eta;
      cout <<"Citta' = ";
      cin >>v[i]->citta;
   }
}

/*una funzione che riceve come parametro formale un vettore di puntatori al tipo
persona, la sua dimensione e calcola l'età più alta tra tutte le variabili i cui
puntatori sono contenuti nel vettore. La funzione deve restituire tale valore.*/

unsigned int max (persona ** v, int dim) {

      unsigned int m=0;
      for (int i=0; i<dim;i++)
            if (v[i]->eta>m)
                        m=v[i]->eta;
```

```
      return m;
}

/* una funzione o procedura che disalloca un vettore di puntatori al tipo
persona.*/

void disalloca(persona **v)
{
      for (int i=0; i<dim;i++)
            delete v[i];
      delete v;
}

/*un main che richiama opportunamente tutte le funzioni precedenti*/

void main()
{
  do {
     cout <<"Inserisci la dimensione ";
     cin>>dimensione;
   } while (dimensione <=0);
   alloca(vettore,dimensione);
   riempi(vettore,dimensione);
   cout<<"L'eta' più alta e' --> "<<max(vettore, dimensione);
   getchar();
   disalloca(vettore);
}
```

## Esempio di Template: Ricerca Binaria

```
#include<iostream.h>
#include<stdio.h>


int *vet_int, elemento_int;
double *vet_double, elemento_double;

int i,dim;

template <class T>
        void inserisci(T v[], int d)
        {
          int i;

          i=0;
          cout << "Inserisci Elemento di indice " << i << " del vettore ";
          cin >> v[i];
          for (i=1;i<d; i++)
                do {
                   cout << "Inserisci Elemento di indice " << i << " del vettore ";
                   cin >> v[i];
                } while (v[i]< v[i-1]);
        }
```

```
template <class T>
        int ricerca_binaria(T v[], T elem, int inf, int sup)
        {
          int medio;

          if (inf < sup) {
            medio=(inf+sup)/2;
            if (v[medio]==elem) return(1);
            else if (v[medio]<elem) return(ricerca_binaria(v,elem,medio+1,sup));
                    else return(ricerca_binaria(v,elem,inf, medio-1));
          } else return(0);
        }


void main()
{
  do {
    cout << "Inserisci la dimensione del vettore ";
    cin >> dim;
  } while (dim<=0);

  vet_double=new double[dim];

  inserisci(vet_double,dim);
  cout << "Inserisci elemento da cercare ";
  cin >> elemento_double;

  if (ricerca_binaria(vet_double,elemento_double,0,dim-1))
              cout << "Elemento trovato " << endl;
  else cout << "Elemento non esistente " << endl;

  delete vet_double;

  do {
    cout << "Inserisci la dimensione del vettore ";
    cin >> dim;
  } while (dim<=0);

  vet_int=new int[dim];

  inserisci(vet_int,dim);
  cout << "Inserisci elemento da cercare ";
  cin >> elemento_int;

  if (ricerca_binaria(vet_int,elemento_int,0,dim-1))
              cout << "Elemento trovato " << endl;
  else cout << "Elemento non esistente " << endl;

  delete vet_int;
```

```
  getchar();
}
```

## Esercizio su ADT Lista

Una segreteria studenti mantiene l'archivio degli studenti iscritti, tramite un vettore a N dimensioni. Ciascun elemento del vettore contiene un puntatore ad una lista di elementi. Ciascun elemento contiene le seguenti informazioni:

Cognome
Nome
Matricola
Residenza
Telefono

Si suppone che ciascuna lista sia ordinata per COGNOME e NOME.

Si suppone che tutte le operazioni tipiche di un archivio (ad esempio inserimento e ricerca) vengano fatte per numero di matricola. Supposto sia m una variabile contenente la matricola dello studente, la somma delle cifre contenute in questa variabile modulo N, fornisce l'indice del vettore in cui effettuare ad esempio l'inserimento e la ricerca.

Si codifichi in C++:

- le strutture dati
- una procedura di inserimento di un nuovo studente. L'inserimento avviene per matricola al fine di individuare la lista in cui effettuare l'inserimento. Una volta individuata la lista, l'inserimento in essa è ORDINATO per COGNOME E NOME.
- una procedura di modifica solo di alcuni campi di uno studente già inserito nell'archivio: residenza e telefono. Lo studente è individuato dalla matricola.
- una procedura di ricerca per matricola, che visualizzi cognome, nome, residenza, e telefono.
- un main che richiami le precedenti procedure

```
#include <iostream.h>
#include <string.h>

/*LISTA PRINCIPALE */

#define LISTAVUOTA NULL

typedef struct studente{
        char cognome[20], nome[20];
        char matricola[10];
        char residenza[30];
        char telefono[15];
} tipobase;

typedef struct nodo {
         tipobase info;
```

```
        struct nodo *next;
}* list;

typedef list position;

void MAKENULL(list &l)
{
  l=LISTAVUOTA;
}

position FIRST(list l)
{
  return(LISTAVUOTA);
}

position END(list l)
{
 if (l==LISTAVUOTA) return(l);
 while (l->next!=LISTAVUOTA)
                        l=l->next;

 return(l);
}

int EMPTY(list l)
{
 return(l==LISTAVUOTA);
}

position LOCATE(list l, tipobase x)
{

 if (!strcmp(l->info.matricola,x.matricola))
                        return(LISTAVUOTA);
 while (l->next!=LISTAVUOTA) {
        if (!strcmp(l->next->info.matricola,x.matricola))
                                return(l);
        l=l->next;
 }
 return(l);
}

void INSERT (list &l, position p, tipobase x)
{
  list temp;

  temp=new struct nodo;
  temp->info=x;

  if (p==LISTAVUOTA) {
        temp->next=l;
```

```
            l=temp;
  } else {
            temp->next=p->next;
            p->next=temp;
  }
}


tipobase RETRIEVE(list l, position p)
{
           if (p==LISTAVUOTA) return(l->info);
           else return(p->next->info);

}

position NEXT(list l, position p)
{
 if (p==LISTAVUOTA) return(l);
            else return(p->next);

}

void insord(list &l, tipobase x)
{
  position p;
  tipobase tmp;

  if (EMPTY(l)) INSERT(l,FIRST(l),x);
  else {
       p=FIRST(l);
       while (p!=END(l)) {
         tmp=RETRIEVE(l,p);
         if (strcmp(tmp.cognome, x.cognome)<0 ||
            (!strcmp(tmp.cognome, x.cognome) && strcmp(tmp.nome, x.nome)<0))
                                              p=NEXT(l,p);
         else break;
       }
       INSERT(l,p,x);
  }
}



void modifica(list l, position p, tipobase x)
{
  if (p==LISTAVUOTA) l->info=x;
  else p->next->info=x;
}

/*Procedura per l'individuazione dell'elemento del vettore */
/*dim e' la dimensione del vettore, e m e' la matricola */
```

```
int calcola_elemento (int dim, char m[])
{
  long int tmp=0;

  for (int i=0; m[i]!='\0'; i++)
       tmp+=m[i];
  return(tmp % dim);
}

/*Procedura per l'inserimento di un nuovo studente*/

void inserisci_studente (list v[], int dim, tipobase x)
{
    int index=calcola_elemento(dim,x.matricola);
    position pos_stud;

    if (!EMPTY(v[index])) pos_stud=LOCATE(v[index], x);

    if (!EMPTY(v[index]) && pos_stud!=END(v[index]))
         cout << "Lo studente e' gia' in archivio " << endl;
    else insord(v[index], x);

}


/*Procedura per la modifica dei dati di uno studente*/

void modifica_dati(list v[], int dim, tipobase x)
{
  int index=calcola_elemento(dim, x.matricola);
  position p;
  char r;

  if (!EMPTY(v[index])) p=LOCATE(v[index], x);

  if (!EMPTY(v[index]) && p!=END(v[index])) {
                      x=RETRIEVE(v[index], p);
                      cout << " Cognome = " << x.cognome << endl;
                      cout << " Nome = " << x.nome << endl;
                      cout << " Residenza = " << x.residenza << endl;
                      cout << " Telefono = " << x.telefono << endl;
       cout << " Vuoi Cambiare la Residenza ?  (s/n)";
       cin >> r;
       if (r=='s' || r=='S') {
                     cout << "Inserisci la nuova residenza ";
            cin >> x.residenza;
       }
       cout << " Vuoi Cambiare il Telefono ?  (s/n)";
       cin >> r;
       if (r=='s' || r=='S') {
```

```
                    cout << "Inserisci il Nuovo Telefono ";
          cin >> x.telefono;
       }
       modifica(v[index],p,x);
   } else cout << "Lista Vuota o Studente non esistente " << endl;
}

/*Procedura per la ricerca e visualizzazione di tutti i dati dello studente */


void ricerca_visualizza(list v[], int dim, tipobase x)
{
  int index=calcola_elemento(dim, x.matricola);
  position p;

  if (!EMPTY(v[index])) p=LOCATE(v[index], x);

  if (!EMPTY(v[index]) && p!=END(v[index])) {
                    x=RETRIEVE(v[index], p);
                    cout << " Cognome = " << x.cognome << endl;
                    cout << " Nome = " << x.nome << endl;
                    cout << " Residenza = " << x.residenza << endl;
                    cout << " Telefono = " << x.telefono << endl;
   } else cout << "Lista Vuota o Studente non esistente " << endl;
}


void main()
{
 int i,s;

 const int N=10;
 list archivio[N];

 tipobase studente;

 for (i=0; i<N; i++)
            MAKENULL(archivio[i]);


 do {
   cout << "Menu di Operazioni" << endl;
   cout << "1-Inserimento Studente " << endl;
   cout << "2-Modifica Dati Studente " << endl;
   cout << "3-Ricerca Studente e Visualizzazione Dati Studente " << endl;
   cout << "4-Fine" << endl;
   cout << "Inserisci la scelta ";
   cin >> s;
   switch(s) {

     case 1 : {
```

```
                cout << "Inserisci i dati dello Studente " << endl;
                cout << "Cognome dello Studente ";
                cin >> studente.cognome;
                cout << "Nome dello Studente ";
                cin >> studente.nome;
                cout << "Residenza dello Studente ";
                cin >> studente.residenza;
                cout << "Telefono dello Studente ";
                cin >> studente.telefono;
                cout << "Matricola dello Studente ";
                cin >> studente.matricola;

        inserisci_studente(archivio,N,studente);
                break;
                }
    case 2 : {

                cout << "Inserisci la Matricola dello Studente ";
                cin >> studente.matricola;

        modifica_dati(archivio,N,studente);

                break;
                }
    case 3 : {
                cout << "Inserisci la Matricola dello Studente ";
                cin >> studente.matricola;

                ricerca_visualizza(archivio, N, studente);
        break;
                }
  }
 } while (s<4);
}
```